interpreted. In order to accomplish this, instrumentation module 12′ may maintain a history of data values that are within the instrumentation routine 35′, 35′′, etc., and derive a unique identifier for each data value that will appear in the trace buffer 103 to facilitate translation of the data into a desired format. The format information for the data is typically contained within the debug information of the application.

The identification of address verses data in the acquired trace may be performed by an extra delimiter value inserted for each operation (instruction branch, or data write) along with the information. Multiple operations of similar type may be associated with a single delimiter. Multiple instruction branch indications may be associated with a single "code" delimiter to save space in the trace buffer. Multiple data values may vary from this model due to the fact that each variable may need to be uniquely identified. In the case of data, there may be a "data" delimiter for each variable in the trace with multiple data values within a structure identified under a single data delimiter. This approach may advantageously serve to reduce the amount of delimiter data required in the trace to convey the information to the host program.

Having thus described the invention, what is claimed is:

CLAIMS

1. A method for monitoring run time execution of software code in a target system, said method comprising:
(a) searching a range of addresses within the software code to identify a desired instruction;
(b) replacing the desired instruction with a program flow change instruction directing execution to a buffer; and
(c) inserting a routine into the buffer, the routine having an output instruction and a branch instruction branching to an address of the software code subsequent to the program flow change instruction.

2. The method of claim 1, wherein the output instruction generates output to a trace buffer.

3. The method of claim 2, wherein the trace buffer is disposed on the target system.

4. The method of claim 1, comprising (e) storing information in an instrumentation table to undo said replacing (b) and said inserting (c).

5. The method of claim 4, wherein the instrumentation table is disposed on a host system communicably coupled to the target system.

6. The method of claim 4, wherein said storing (e) comprises storing the desired instruction, address of

the desired instruction, action to be performed by the program flow change instruction, address of the buffer, size of the routine, and an identifier associated with the action to be performed.

7. The method of claim 1, wherein the target system includes a cache and at least a portion of the software code executes externally of the cache.

8. The method of claim 1, wherein the target system includes a bus and at least a portion of the software code executes on the bus.

9. The method of claim 1, wherein said searching (a) further comprises searching for a plurality of desired instructions.

10. The method of claim 1, wherein the routine comprises a cache disabling instruction and a cache re-enabling instruction.

11. The method of claim 1, wherein said searching (a) comprises searching for a desired instruction disposed at the beginning of a program function.

12. The method of the claim 11, wherein the desired instruction comprises a Move From Special Register instruction.

13. The method of claim 11, wherein said searching (a) comprises searching for an other desired instruction disposed at the ending of a program function.

14. The method of claim 13, wherein the other desired instruction comprises a Move To Special Register instruction.

15. The method of claim 1, wherein said searching (a) comprises searching for at least one desired instruction associated with data manipulation.

16. The method of claim 1, wherein said inserting (c) comprises inserting a routine having a Data Cache-disabling instruction.

17. The method of claim 1, wherein said inserting (c) comprises inserting a routine having an Instruction Cache-disabling instruction.

18. The method of claim 1, wherein said searching (a) comprises searching for a branch instruction, and searching for the desired instruction in a portion of the software code indicated by the branch instruction, the desired instruction being disposed outside of the range of addresses identified.

19. The method of claim 1, wherein the desired instruction comprises an EABI instruction.

20. The method of claim 1, wherein the searching (a) comprises using debug information to identify the desired instruction.

21. The method of claim 20, wherein the searching (a) comprises using complier-derived debug information in a format selected from the group consisting of stabs, elf, and dwarf formats.

22. The method of claim 1, wherein the program flow change instruction comprises an instruction to read from an odd address.

23. The method of claim 22, wherein the program flow change instruction comprises an instruction to add an odd integer to an address.

24. The method of claim 23, wherein the routine has a decoding instruction to identify the odd integer and execute an instruction corresponding thereto.

25. The method of claim 1, comprising a plurality of program flow change instructions corresponding to a plurality of user-selectable operations.

26. The method of claim 25, wherein each of said plurality of user-selectable operations is selected from the group consisting of:

indicating entry and exit of a function;

indicating entry and exit of a function and tracing execution of a function;

indicating entry and exit of a function, tracing execution of the function, and indicating entry and exit and tracing execution of other functions called by the function;

indicating Entry and Exit of a function, tracing execution of the function, and indicating Entry and Exit without tracing execution of other functions called by the function;

indicating data manipulation;

inserting patch code into a code portion;

indicating the sequence of program execution; and

indicating changes to variables.

27. The method of claim 26, wherein said inserting (c) comprises:

(i) selecting at least one output code statement to perform a selected one of said user-selectable operations;

(ii) saving a copy of the output code statement and the desired instruction;

(iii) determining the size of the output code statement, the branch instruction, the desired instruction, and restore code to restore the desired instruction; and

(iv) allocating memory in the buffer of the size determined in (g); and

(v) inserting the output code statement, the branch instruction, the desired instruction, and restore code, into the allocated memory.

28. The method of claim 27, wherein said saving (ii) comprises saving a copy of the program flow change instruction and the desired instruction in a translation table.

29. The method of claim 27, wherein said selecting

(i) comprises analyzing a symbol table of the software code.

30. The method of claim 27, wherein said selecting (i) comprises calling a function selected from the group consisting of a printf or scanf function.

31. The method of claim 27, wherein the restore code comprises code to save and restore original register contexts.

32. The method of claim 26, wherein said searching (a) comprises identifying addresses in the program code that are associated with each instance of a modification of an identified variable/structure, and locating a final instruction for each instance of a modification, the final instruction being said desired instruction.

33. The method of claim 32, wherein said inserting (c) comprises:

(i) selecting at least one output code statement to transfer data to the buffer;

(ii) saving a copy of the output code statement and the desired instruction;

(iii) determining the size of the output code statement, the desired instruction, and restore code to restore the desired instruction;

(iv) allocating memory in the buffer of the size determined in (g), and to run the trace acquisition code;

(v) inserting the output code statement, the branch instruction, the desired instruction, and restore code, into the allocated memory.

34. The method of claim 33, wherein said allocating (iv) further comprises allocating additional memory of the size determined in (iii) for each said instance of a modification of an identified variable/structure.

35. The method of claim 34, further comprising repeating said inserting (v) for each said instance.

36. The method of claim 33, wherein said saving (ii) comprises saving a copy of the program flow change instruction and the desired instruction in a translation table.

37. The method of claim 33, wherein said selecting (i) comprises analyzing a symbol table of the software code.

38. The method of claim 33, wherein said selecting (i) comprises calling a function selected from the group consisting of a printf or scanf function.

39. The method of claim 33, wherein the restore code comprises code to save and restore original register contexts.

40. The method of claim 1, further comprising reversing said replacing (b), and inserting (c), to restore the software code.

41. The method of claim 1, wherein at least one of said searching (a), replacing (b), and inserting (c), is performed during run time execution of the software code.

42. The method of claim 41, wherein at least one of said searching (a), replacing (b), and inserting (c), is performed after the software code is compiled.

43. The method of claim 42, wherein execution of the software code is halted during performance of said at least one of said searching (a), replacing (b), and inserting (c).

44. The method of claim 1, comprising executing the software code.

45. A method for monitoring run time execution of software code in a target system, said method comprising:
searching a range of addresses within the software code using debug information to identify a desired instruction;
replacing the desired instruction with a program flow change instruction directing execution to a buffer; and
inserting a routine into the buffer, the routine having a branch instruction branching to an address of the software code subsequent to the program flow change instruction.

46. A system for monitoring run time execution of software code in a target, said system comprising:

an instruction locating module configured to search a range of addresses within the software code to identify a desired instruction;

an instruction replacement module configured to replace the desired instruction with a program flow change instruction directing execution to a buffer; and

an instrumentation module configured to insert a routine into the buffer, the routine having an output instruction and a branch instruction branching to an address of the software code subsequent to the program flow change instruction.

47. A system for monitoring run time execution of software code in a target, said system comprising:

an instruction locating module configured to search a range of addresses within the software code using debug information to identify a desired instruction;

an instruction replacement module configured to replace the desired instruction with a program flow change instruction directing execution to a buffer; and

an instrumentation module configured to insert a routine into the buffer, the routine having a branch instruction branching to an address of the software

code subsequent to the program flow change instruction.

48. An article of manufacture for monitoring run time execution of software code in a target system, said article of manufacture comprising:

a computer usable medium having computer readable program code embodied therein, said computer usable medium having:

computer readable program code for searching a range of addresses within the software code to identify a desired instruction;

computer readable program code for replacing the desired instruction with a program flow change instruction directing execution to a buffer; and

computer readable program code for inserting a routine into the buffer, the routine having an output instruction and a branch instruction branching to an address of the software code subsequent to the program flow change instruction.

49. Computer readable program code for monitoring run time execution of software code in a target system, said computer readable program code comprising:

computer readable program code for searching a range of addresses within the software code to identify a desired instruction;

computer readable program code for replacing the desired instruction with a program flow change instruction directing execution to a buffer; and

computer readable program code for inserting a routine into the buffer, the routine having an output instruction and a branch instruction branching to an address of the software code subsequent to the program flow change instruction.